GENERALISED TRANSFER FUNCTIONS OF NEURAL NETWORKS

C. F. Fung†, S. A. Billings† and H. Zhang‡

†Department of Automatic Control
and Systems Engineering
University of Sheffield, Mappin Street
Sheffield S1 3JD

‡Predictive Control Limited,
Richmond House
Gadbrook Business Centre, Northwich
Cheshire CW9 7TN

22 May, 1996

Research Report #627

Contents

Abstract

When artificial neural networks are used to model nonlinear dynamical systems, the system structure which can be extremely useful for analysis and design, is buried within the network architecture. In this paper explicit expressions for the frequency response or generalised transfer functions of both feedforward and recurrent neural networks are derived in terms of the network weights. The derivation of the algorithm is established on the basis of the Taylor series expansion of the activation functions used in a particular neural network. This leads to a representation which is equivalent to the nonlinear recursive polynomial model and enables the derivation of the transfer functions to be based on the harmonic expansion method. By mapping the neural network into the frequency domain information about the structure of the underlying nonlinear system can be recovered. Numerical examples are included to demonstrate the application of the new algorithm. These examples show that the frequency response functions appear to be highly sensitive to the network topology and training, and that the time domain properties fail to reveal deficiencies in the trained network structure.

Keywords

Generalised transfer function, generalised frequency response function, frequency domain analysis, multilayered perceptron network, system identification, nonlinear dynamical system modelling.

GENERALISED TRANSFER FUNCTIONS OF NEURAL NETWORKS

Chi F. Fung†, Steve A. Billings† & Huaiqiang Zhang‡

†Department of Automatic Control & Systems Engineering
University of Sheffield, Mappin Street, Sheffield S1 3JD
United Kingdom.

Telephone: +44(0)114 273 5232
Facsimile: +44(0)114 276 1729

‡Predictive Control Limited
Richmond House, Gadbrook Business Centre
Northwich, Cheshire CW9 7TN
United Kingdom

Telephone: +44(0)1606 44626
Facsimile: +44(0)1606 44592

22 May, 1996

Abstract

When artificial neural networks are used to model nonlinear dynamical systems, the system structure which can be extremely useful for analysis and design, is buried within the network architecture. In this paper explicit expressions for the frequency response or generalised transfer functions of both feedforward and recurrent neural networks are derived in terms of the network weights. The derivation of the algorithm is established on the basis of the Taylor series expansion of the activation functions used in a particular neural network. This leads to a representation which is equivalent to the nonlinear recursive polynomial model and enables the derivation of the transfer functions to be based on the harmonic expansion method. By mapping the neural network into the frequency domain information about the structure of the underlying nonlinear system can be recovered. Numerical examples are included to demonstrate the application of the new algorithm. These examples show that the frequency response functions appear to be highly sensitive to the network topology and training, and that the time domain properties fail to reveal deficiencies in the trained network structure.

Keywords

Generalised transfer function, generalised frequency response function, frequency domain analysis, multilayered perceptron network, system identification, nonlinear dynamical system modelling.

1. Introduction

Artificial neural networks (ANN's) have been widely accepted as a powerful alternative tool for modelling nonlinear dynamical systems and provide an alternative to existing nonlinear identification techniques such as the Volterra series and NARMAX (*nonlinear auto-regressive moving average model with exogenous inputs*) methodologies. Numerous authors have studied the modelling and control of nonlinear dynamical systems using ANN's [18][16]. However, several authors [6] have pointed out that although ANN's can give excellent predictions after proper training, one disadvantage of using ANN's to model nonlinear systems is that they tend to destroy the system structure which can be important for system design and control. In the extreme case, if the underlying system happens to be linear, and an ANN is blindly used to model the system, the fact that the system is linear will not be evident after the network has been trained. In this case the system can be modelled using a simple pulse transfer function model and analysis and control can be achieved using conventional linear systems theory. But this will not be apparent from the trained ANN which is likely to be a nonlinear model fitted to represent a linear dynamic relationship. Similar scenarios can also arise in nonlinear systems where typical nonlinearities such as square or cubic law devices could be physically meaningful to the understanding of the systems but which will not be evident in the trained ANN model. Although the concept of parsimonious models is fundamentally important in the context of system modelling, unfortunately this issue has been largely ignored in the majority of neural network applications. The potential recovery of structural information buried within a trained network model has rarely been addressed.

The impulse response function is a standard model for characterising linear systems in classical control theory. The advantage of using the impulse response is that this gives a concise quantitative description of the system in the time domain which can easily be transformed into an equivalent frequency domain transfer function description. Furthermore, both the impulse response and the transfer function descriptions are independent of the input signal so that they provide a universally valid mathematical description which can be used for system analysis and design.

When these ideas are extended to nonlinear system analysis, the usefulness of the impulse response or transfer function description depends critically on the way the nonlinear system is represented. One of the most commonly used representations in nonlinear modelling and identification is the Volterra series [22] which expresses the system response in terms of generalised impulse response functions or Volterra kernels. By using the Volterra series representation the advantages of the linear impulse response concepts can be carried over to the nonlinear case. The equivalent nonlinear frequency domain representation can also be obtained by

applying the (multivariate) Fourier transform to the Volterra kernels to yield the *generalised transfer functions* (GTF's) or *generalised frequency response functions* (GFRF's). Of course, the Volterra series representation has its limitations see for example Wray and Green [27].

In practical identification, Volterra series representations are rarely used because of the well known curse-of-dimensionality problem and the practical computational difficulties associated with the estimation of the Volterra kernels [27]. Polynomial NARMAX model (nonlinear autoregressive moving average model with exogenous inputs) representations are found to be significantly more economical in the number of terms required because delayed output variables are included in the representation. As a result, NARMAX representations have been widely used in nonlinear system identification [3][2].

Although the usefulness of Volterra series representations are in practice largely confined to theoretical analysis, the idea of interpreting the Volterra kernels as impulse response functions can effectively be extended to polynomial NARMAX models. Such an extension is important in the derivation of the GFRF's for polynomial NARMAX models via the *harmonic expansion method* which has been documented in [8].

When artificial neural networks are used to model a nonlinear system, the GFRF's become particularly important in the context of information recovery. The GFRF technique, when properly interpreted, can be taken as an auxiliary validation tool to minimise the risk of obtaining an under- or over-fitted network model. The basic principle of the derivation of the GFRF's for an ANN is motivated by the fact that a functional equivalence can be established between the underlying network model and the Volterra series representation as shown by Wray and Green [27]. It can therefore be shown that techniques similar to those used to compute the GFRF's for Volterra series and polynomial NARX† models can also be applied to the derivation of GFRF's for neural network models [27].

In the present study GFRF's are derived for both feedforward and recurrent networks in an attempt to reveal useful information about the underlying system. The analysis is focused upon the derivation of GFRF's for three layered perceptrons because of the simplicity and relevance of this architecture in system modelling. Analytical expressions are derived to relate the GFRF's to the weights for networks of this form and simulated examples are used to demonstrate the application of the results. The examples appear to suggest that the reconstruction of the GFRF's for neural networks are extremely sensitive to slight changes in the network topology and the training procedure.

---

† NARX refers to a NARMAX model where the MA or noise terms have been discarded.

2. Generalised Frequency Response Functions for Nonlinear Systems

It is a well known fact that for a single-input, single output, analytic continuous time invariant causal system, the system input $u(t)$ and the system output $y(t)$ at time t can be related by

$$y(t)=h_0+\int_{0-}^{\infty}h_1(\tau_1)u(t-\tau_1)\mathrm{d}\tau_1+\int_{0-}^{\infty}\int_{0-}^{\infty}h_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)\mathrm{d}\tau_1\mathrm{d}\tau_2+\cdots$$
$$\cdots+\int_{0-}^{\infty}\cdots\int_{0-}^{\infty}h_n(\tau_1,\cdots,\tau_n)u(t-\tau_1)\cdots u(t-\tau_n)\mathrm{d}\tau_1\cdots\mathrm{d}\tau_n+\cdots \quad (1)$$

where $h_n(\tau_1,\cdots,\tau_n)\in\mathbb{R}$ is the n-th order *Volterra kernel*. The system representation given by equation (1) is generally referred to as the Volterra series [22]. Notice that the constant term h_0 has been deliberately included as a *dc* offset since this often affects the dynamics of nonlinear systems [8]. Alternatively, equation (1) can be written in the following form

$$y(t)=y_0+y_1(t)+\cdots+y_n(t)+\cdots \quad (2)$$

where $y_n(t)$ is the n-th order output which is produced by the n-th order *Volterra operator* $\mathcal{V}_n(\cdot)$ defined by

$$y_n(t)=(\mathcal{V}_n u)(t)$$
$$\triangleq\begin{cases}h_0 & \text{if } n=0\\ \int_{0-}^{\infty}\cdots\int_{0-}^{\infty}h_n(\tau_1,\cdots,\tau_n)u(t-\tau_1)\cdots u(t-\tau_n)\mathrm{d}\tau_1\cdots\mathrm{d}\tau_n & \text{if } n\geq 1\end{cases} \quad (3)$$

Inspection of equation (1) suggests that the Volterra kernel, $h_n(\tau_1,\cdots,\tau_n)$, can be regarded as the n-th order generalisation of the conventional impulse response function associated with a linear system. A wide class of nonlinear systems encountered in science and engineering can be modelled by the Volterra series and this model has been used in many applications. A system with a Volterra series representation can be described, equivalently, in the frequency domain by applying the multivariate Fourier transform to the Volterra kernel $h_n(\tau_1,\cdots,\tau_n)$. Define the Fourier transform of $h_n(\tau_1,\cdots,\tau_n)$, $H_n(f_1,\cdots,f_n)\in\mathbb{C}$, as

$$H_n(f_1,\cdots,f_n)\triangleq\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}h_n(\tau_1,\cdots,\tau_n)\mathrm{e}^{-\mathrm{j}2\pi(f_1\tau_1+\cdots+f_n\tau_n)}\mathrm{d}\tau_1\cdots\mathrm{d}\tau_n,\quad n\geq 1 \quad (4)$$

where $\mathrm{j}=\sqrt{-1}$. This alternative description characterised by $H_n(f_1,\cdots,f_n)$ is commonly called the n-th order generalised frequency response function or sometimes the generalised transfer function. Notice that equation (4) is consistent with the conventional definition of standard transfer functions

and reduces to $H_1(f_1)$ for the case when $n = 1$. The importance of the higher order transfer function has been recognised since the early sixties [10][11] because of applications in the frequency domain analysis of nonlinear systems. Both $h_n(\tau_1,\cdots,\tau_n)$ and $H_n(f_1,\cdots,f_n)$ provide an invariant description which is independent of the system excitation. Indeed, since the n-th order impulse response (or Volterra kernel) $h_n(\tau_1,\cdots,\tau_n)$ and n-th order GFRF $H_n(f_1,\cdots,f_n)$ are Fourier transform pairs. The n-th order system output, $y_n(t)$, defined in equation (3) can also be written as

$$y_n(t) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} H_n(f_1,\cdots,f_n) \prod_{i=1}^{n} U(f_i) e^{j2\pi(f_1+\cdots+f_n)t} df_1 \cdots df_n, \qquad n \geq 1 \tag{5}$$

where $U(f_i)$ denotes the Fourier spectrum of the system input $u(t)$.

Based on the properties of the Fourier transform, the GFRF's in equation (4) can be adopted for both continuous and discrete time cases. In the present study, all the systems under investigation will be discrete time physically realisable systems with a Volterra series representation given by the discrete time equivalent to equation (1)†

$$y(k) = h_0 + \sum_{k_1=0}^{\infty} h_1(k_1)u(k-k_1) + \cdots + \sum_{k_1,\cdots k_n=0}^{\infty} h_n(k_1,\cdots,k_n) \prod_{i=1}^{n} u(k-k_i) + \cdots \tag{6}$$

where $u(k)$ and $y(k)$ are the (sampled) system input and output variables at time step k, respectively. Boyd and Chua [9] have rigorously proved that any finite memory nonlinear system with memory length, say N samples, can be represented, with arbitrary degree of accuracy, by a truncated Volterra series. Therefore, a system with finite memory can be expressed as

$$y(k) = h_0 + \sum_{k_1=0}^{M} h_1(k_1)u(k-k_1) + \cdots + \sum_{k_1,\cdots k_N=0}^{M} h_N(k_1,\cdots,k_N) \prod_{i=1}^{N} u(k-k_i) \tag{7}$$

where $M < \infty$. Finite term Volterra series have been found to be extremely useful in modelling many practical nonlinear dynamical systems in diverse disciplines ranging from electrical engineering to physiology.

For a discrete time system with a Volterra series representation given by equation (7), the n-th order GFRF can be defined by taking the (multivariate) z-transform of the Volterra kernel, $h_n(k_1,\cdots,k_n)$ to yield‡

---

† The n-tuple summation is denoted as $\Sigma_{k_1,\cdots,k_n=1}^{\infty}(\cdot) \equiv \Sigma_{k_1=1}^{\infty}\cdots\Sigma_{k_n=1}^{\infty}(\cdot)$. The same notional convention applies throughout.

‡ From this point onwards, all frequency variables will be assumed to be normalised to their respective sampling frequencies by adopting the principal values in [-1/2, 1/2].

$$H_n(f_1,\cdots,f_n)\triangleq\sum_{k_1,\cdots k_n=0}^{N} h_n(k_1,\cdots,k_n)e^{-j2\pi(f_1k_1+\cdots+f_nk_n)},\qquad n\geq 1 \tag{8}$$

In contrast to the definition given in equation (4) for continuous time systems, non-negative time indices have been used to enforce system causality in equation (8).

3. Generalised Frequency Response Functions for Feedforward Networks

To derive the transfer function for ANN's, consider initially a *multilayered perceptron* (MLP) in the simplest form as depicted in Figure 1. It is a well known fact that a MLP with a single hidden layer can approximate arbitrarily well any continuous function on a compact support provided that a sufficient number of hidden units are included [13]. The network shown in Figure 1 consists of an input layer of m input units, a hidden layer of n_h units and an output layer of one unit. At time step k, the m input units receive input signals $\{x_j(k): 1\leq j\leq m\}$. These input signals are weighted and summed appropriately, according to a set of connection weights to produce the input to the i-th hidden unit, $\upsilon_i(k)$, such that

$$\upsilon_i(k)=\theta_{i,1}x_1(k)+\theta_{i,2}x_2(k)+\cdots+\theta_{i,m}x_m(k),\qquad i\in\{1,2,\cdots,n_h\} \tag{9}$$

where $\theta_{i,j}$ $(1\leq j\leq m)$ denotes the connection weight between the i-th hidden unit and the j-th input unit. Thus, the output of the i-th hidden unit, $\varphi_i(k)$, is obtained by mapping $\upsilon_i(k)$ together with the bias term b_i via a nonlinear *activation function* $\sigma:\mathbb{R}\rightarrow\mathbb{R}$ which is commonly taken as a sigmoid or

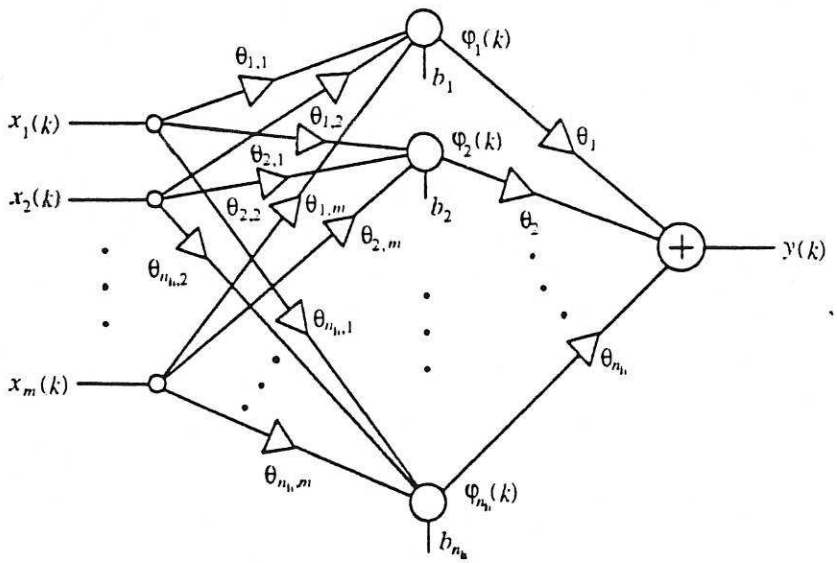


Figure 1 m-input, single output three-layered perceptron.

hyperbolic tangent function to yield

$$\varphi_i(k) = \sigma(\upsilon_i + b_i), \qquad i \in \{1, 2, \cdots, n_h\} \tag{10}$$

At the output layer, the network output, $y(k)$, is obtained by taking the weighted sum of these hidden layer outputs according to

$$y(k) = \theta_1\varphi_1(k) + \theta_2\varphi_2(k) + \cdots + \theta_{n_h}\varphi_{n_h}(k) \tag{11}$$

where θ_i denotes the connection weight between the output unit and the i-th hidden unit. Equation (11) can therefore be expanded to yield

$$y(k) = \sum_{i=1}^{n_h} \theta_i \sigma\left(\theta_{i,1}x_1(k) + \theta_{i,2}x_2(k) + \cdots + \theta_{i,m}x_m(k) + b_i\right) \tag{12}$$

For the sake of efficient training, the activation function, $\sigma(\cdot)$, is usually chosen to be analytic. Based on the well known Weierstrass' theorem, it is realistic to assume that $\sigma(\cdot)$ can be approximated, uniformly, with arbitrary degree of accuracy, by an algebraic polynomial on a closed interval. In fact, some ANN's simply use polynomials as their activation functions [26][27]. Of course, these types of network have poorer function approximation properties since they can only produce a Volterra series within a finite number of terms [27]. By adopting this point of view, it will be assumed that the activation function can be approximated by

$$\sigma(\upsilon_i + b_i) \approx \psi_i^{(0)} + \psi_i^{(1)}\upsilon_i + \cdots + \psi_i^{(r)}\upsilon_i^r + \cdots + \psi_i^{(P)}\upsilon_i^P \tag{13}$$

where P is a sufficiently large positive integer which gives rise to the maximum degree of the polynomials in the expansion of the activation function. Notice that the bias term b_i has been incorporated into the polynomial coefficients. For a given activation function, $\sigma(\cdot)$, the coefficient of the r-th degree term, $\psi_i^{(r)}$, in the polynomial can be determined by a Taylor expansion of the activation function about the bias such that

$$\psi_i^{(r)} \triangleq \frac{1}{r!}\frac{\mathrm{d}^r\sigma}{\mathrm{d}\upsilon_i^r}\bigg|_{\upsilon_i = b_i}, \qquad r = 0, 1, 2, \cdots, P \tag{14}$$

Combining equation (12) and equation (13) gives

$$y(k) = \sum_{i=1}^{n_h} \theta_i \sum_{r=0}^{P} \psi_i^{(r)} \left[\sum_{j=1}^{m} \theta_{i,j}x_j(k)\right]^r \tag{15}$$

Before the derivation can proceed further it is important to note that in order to maintain the validity of equation (15), which subsequently plays a crucial part in determining the final form of the GFRF's for both feedforward and recurrent networks, the input signals to the network must satisfy the convergence conditions required for the Taylor series expansion of the activation

function. In other words, the Taylor series represented by equation (13) must be convergent as $P \to \infty$. Obviously, convergence depends on the particular activation function being employed. Taking the most commonly used hyperbolic tangent as an example, the Taylor series expansion requires the range of input signal to be within the radius of convergence in $(-\pi/2, \pi/2)$ or $|\upsilon_i| < \pi/2$. This highlights one of the limitations in expanding an ANN in terms of a Taylor series. Although this condition can easily be met in typical network design it does not universally hold in general. When this condition is violated, equation (15) will no longer be valid and consequently, an incorrect expression for the GFRF will result.

When a feedforward network is employed in dynamical system modelling, the input nodes are often assigned to be the delayed samples of the system inputs, $u(k-1), u(k-2), \cdots$. Such a network structure is referred to as a *time-delay neural network* [23]. Denote $\mathbf{x}(k) \in \mathbb{R}^m$, as the network input vector whose j-th component, $x_j(k)$, is given by

$$x_j(k) = u(k-j), \qquad j \in \{1, 2, \cdots, m\} \tag{16}$$

Thus,

$$\mathbf{x}(k) \underline{\Delta} [u(k-1), u(k-2), \cdots, u(k-m)]^{\mathrm{T}} \tag{17}$$

equation (15) then becomes

$$y(k) = \sum_{i=1}^{n_h} \theta_i \sum_{r=0}^{P} \psi_i^{(r)} \sum_{p_1, \cdots, p_r = 1}^{m} \prod_{v=1}^{r} \theta_{i,p_v} u(k-p_v) \tag{18}$$

Comparing equation (18) with equation (7) shows that equation (18) can be viewed as a finite memory (with memory length m) discrete time Volterra series. This shows that under the conditions given above a feedforward neural network can be expressed as an equivalent Volterra series [27] in which the n-th order Volterra kernel is given by

$$h_n(p_1, \cdots, p_n) = \begin{cases} \sum_{i=1}^{n_h} \theta_i \sigma(b_i) & \text{if } n = 0 \\ \sum_{i=1}^{n_h} \theta_i \psi_i^{(n)} \prod_{v=1}^{n} \theta_{i,p_v} & \text{if } n \geq 1 \end{cases} \tag{19}$$

Using the definition of equation (8), the n-th order GFRF of equation (19) can be written as

$$H_n(f_1, \cdots, f_n) = \sum_{p_1, \cdots, p_n = 1}^{m} \sum_{i=1}^{n_h} \theta_i \psi_i^{(n)} \prod_{v=1}^{n} \theta_{i,p_v} e^{-j2\pi(f_1 p_1 + \cdots + f_n p_n)}, \qquad n \geq 1 \tag{20}$$

Notice that this includes contributions from, and only from, all the n-degree pure input product terms. The resulting GFRF is a continuous function of the n frequency variables $f_1, f_2, \cdots, f_n$.

Once a multilayered perceptron has been trained all the weights θ_i are known and equation (20) can therefore be used to map the network into the frequency domain.

4. Generalised Frequency Response Functions for Recurrent Networks

A procedure similar to that given in Section 3 for feedforward networks can be used to derive the GFRF's for the case where the network input vector consists of both delayed system inputs and delayed network outputs $y(k-1), y(k-2), \cdots$. For an m-input recurrent network, the network input vector, $\mathbf{x}(k)$, is redefined as

$$\mathbf{x}(k) \triangleq \left[u(k-1), u(k-2), \cdots, u(k-n_u), y(k-1), y(k-2), \cdots, y(k-n_y)\right]^{\mathrm{T}} \tag{21}$$

where n_u and n_y are, respectively, the highest lag in the network input and output variables with $n_u + n_y \equiv m$. In the context of the present study, network structures with delayed feedback outputs as part of the network input are referred to as *recurrent networks* [14][7].

The inclusion of the feedback components in $\mathbf{x}(k)$ means that the derivation of the GFRF's for recurrent networks is more complicated because the extraction of the Volterra kernels is no longer straightforward. In addition, as will become clear later, the presence of non-zero bias terms at each of the hidden nodes generates a non-zero *dc* offset which contributes to the overall network output as a mean level and this dramatically increases the complexity of the analytical expressions for the underlying GFRF's [20][8]. Fortunately, by reformulating the recurrent network in terms of a polynomial NARX model, the task of the derivation can be greatly reduced by utilising similar features from the derivation based on the established polynomial NARX model. Adopting this strategy, the derivation of the GFRF's for recurrent networks can be carried out in two stages.

In the first stage, the network response is decomposed into a number of different groups of signal categories which are classified according to the source of each contribution. With the definition of the network input vector in equation (21), the network output, $y(k)$, can be written as

$$y(k) = \sum_{i=1}^{n_h} \theta_i \sum_{r=0}^{P} \psi_i^{(r)} \left[\sum_{j=1}^{n_u} \theta_{i,j} u(k-j) + \sum_{h=1}^{n_y} \theta_{i,h+n_u} y(k-h) \right]^r \tag{22}$$

Using the binomial theorem, the entry within square brackets can be expanded to yield

$$\left[\sum_{j=1}^{n_u}\theta_{i,j}u(k-j)+\sum_{h=1}^{n_y}\theta_{i,h+n_u}y(k-h)\right]^r=\sum_{l=0}^{r}\binom{r}{l}\left[\sum_{j=1}^{n_u}\theta_{i,j}u(k-j)\right]^{r-l}\left[\sum_{h=1}^{n_y}\theta_{i,h+n_u}y(k-h)\right]^l \quad (23)$$

Likewise, the two power terms on the right hand side of equation (23) can be further expanded to yield

$$\left[\sum_{j=1}^{n_u}\theta_{i,j}u(k-j)\right]^{r-l}=\begin{cases}\displaystyle\sum_{p_1,\cdots,p_{r-l}=1}^{n_u}\prod_{v=1}^{r-l}\theta_{i,p_v}u(k-p_v) & \text{if } 0\le l\le r-1\\ 1 & \text{if } l=r\end{cases} \quad (24)$$

and

$$\left[\sum_{h=1}^{n_y}\theta_{i,h+n_u}y(k-h)\right]^l=\begin{cases}1 & \text{if } l=0\\ \displaystyle\sum_{q_1,\cdots,q_l=1}^{n_y}\prod_{s=1}^{l}\theta_{i,q_s+n_u}y(k-q_s) & \text{if } 1\le l\le r\end{cases} \quad (25)$$

It is clear that equations (24) and (25), respectively, comprise all the possible combinations of $(r-l)$-th degree pure input and l-th degree pure output product terms. These results can now be combined and substituted into equation (23) so that the power of the sum of the input and output components can be expressed in terms of the sum of three signal aggregates as

$$\begin{aligned}\left[\sum_{j=1}^{n_u}\theta_{i,j}u(k-j)+\sum_{h=1}^{n_y}\theta_{i,h+n_u}y(k-h)\right]^r &= \sum_{p_1,\cdots,p_r=1}^{n_u}\prod_{v=1}^{r}\theta_{i,p_v}u(k-p_v)\\ &+\sum_{l=1}^{r-1}\binom{r}{l}\left[\sum_{p_1,\cdots,p_{r-l}=1}^{n_u}\prod_{v=1}^{r-l}\theta_{i,p_v}u(k-p_v)\right]\left[\sum_{q_1,\cdots,q_l=1}^{n_y}\prod_{s=1}^{l}\theta_{i,q_s+n_u}y(k-q_s)\right]\\ &+\sum_{q_1,\cdots,q_r=1}^{n_y}\prod_{s=1}^{r}\theta_{i,q_s+n_u}y(k-q_s)\end{aligned} \quad (26)$$

Replacing the square bracket term in equation (22) with equation (26), the network response can be fully expanded to yield

$$\begin{aligned}y(k)=&\underbrace{\sum_{i=1}^{n_h}\theta_i\sigma(b_i)}_{\text{constant term}}+\underbrace{\sum_{i=1}^{n_h}\theta_i\sum_{r=1}^{P}\psi_i^{(r)}\sum_{p_1,\cdots,p_r=0}^{n_u}\prod_{v=1}^{r}\theta_{i,p_v+1}u(k-p_v)}_{\text{pure input term}}\\ &+\underbrace{\sum_{i=1}^{n_h}\theta_i\sum_{r=2}^{P}\psi_i^{(r)}\sum_{l=1}^{r-1}\binom{r}{l}\left[\sum_{p_1,\cdots,p_{r-l}=0}^{n_u}\prod_{v=1}^{r-l}\theta_{i,p_v+1}u(k-p_v)\right]\left[\sum_{q_1,\cdots,q_l=1}^{n_y}\prod_{s=1}^{l}\theta_{i,q_s+n_u+1}y(k-q_s)\right]}_{\text{cross product term}}\end{aligned}$$

$$\underbrace{+\sum_{i=1}^{n_h}\theta_i\sum_{r=1}^{P}\psi_i^{(r)}\sum_{q_1,\cdots,q_r=1}^{n_y}\prod_{s=1}^{r}\theta_{i,q_s+n_u+1}y(k-q_s)}_{\text{pure output term}} \tag{27}$$

From equation (27), it is clear that the network output is composed of four categories of signal. These are labelled as the *constant term*, *pure input term*, *cross product term* and *pure output term*. Despite the awkward form of the expression the input-output relationship expressed above is in fact a polynomial NARX model [17]. Equation (27) shows that the constant term is solely due to the contribution from the 0-th degree terms, $\psi_i^{(0)}(\equiv\sigma(b_i))\ (i=1, 2, \cdots, n_h)$, in the Taylor series expansion for the activation function. The effects of the presence of constant terms in nonlinear systems has been fully studied elsewhere [20]. By applying the results presented previously in [8], the derivation of n-th order GFRF for a recurrent network based on equation (27) is rather straightforward using the harmonic expansion method [12][4][19]. Such a method has recently been extended and generalised to accommodate the case where a non-zero constant term arises in the time domain model [8][28]. Using the harmonic expansion method, the procedure required to derive the n-th order GFRF involves substituting the harmonic input (which is a sum of complex exponentials)

$$u(k)=e^{j2\pi f_1 k}+e^{j2\pi f_2 k}+\cdots+e^{j2\pi f_n k} \tag{28}$$

into the Volterra series defined in equation (6) to give

$$y(k)=h_0+\sum_{k_1=0}^{\infty}h_1(k_1)\left\{e^{j2\pi f_1(k-k_1)}+e^{j2\pi f_2(k-k_1)}+\cdots+e^{j2\pi f_n(k-k_1)}\right\}+\cdots$$
$$\cdots+\sum_{k_1,\cdots,k_n=0}^{\infty}h_n(k_1,\cdots,k_n)\prod_{i=1}^{n}\left\{e^{j2\pi f_1(k-k_i)}+e^{j2\pi f_2(k-k_i)}+\cdots+e^{j2\pi f_n(k-k_i)}\right\}+\cdots \tag{29}$$

Under the excitation of the harmonic input specified in equation (28), the response of the recurrent network expressed in equation (27) can be obtained by substituting equations (28) and (29) into equation (27) to yield

$$y(k)=\underbrace{\sum_{i=1}^{n_h}\theta_i\sigma(b_i)}_{\text{constant term}}+\underbrace{\sum_{i=1}^{n_h}\theta_i\sum_{r=1}^{P}\psi_i^{(r)}\sum_{p_1,\cdots,p_r=0}^{n_u}\prod_{v=1}^{r}\theta_{i,p_v+1}\sum_{\ell=1}^{n}e^{j2\pi f_\ell(k-p_v)}}_{\text{pure input term}}$$
$$\underbrace{+\sum_{i=1}^{n_h}\theta_i\sum_{r=2}^{P}\psi_i^{(r)}\sum_{l=1}^{r-1}\binom{r}{l}\left\{\sum_{p_1,\cdots,p_{r-l}=0}^{n_u}\prod_{v=1}^{r-l}\theta_{i,p_v+1}\sum_{\ell=1}^{n}e^{j2\pi f_\ell(k-p_v)}\right\}\left\{\sum_{q_1,\cdots,q_l=1}^{n_y}\prod_{s=1}^{l}\theta_{i,q_s+n_u+1}\right\}\left[h_0+\right.}_{\text{cross product term}}$$

$$\underbrace{\left.\sum_{k_1=0}^{\infty} h_1(k_1)\sum_{\ell=1}^{n} e^{j2\pi f_\ell(k-q_s-k_1)} + \cdots + \sum_{k_1,\cdots k_n=0}^{\infty} h_n(k_1,\cdots,k_n)\prod_{i=1}^{n}\sum_{\ell=1}^{n} e^{j2\pi f_\ell(k-q_s-k_i)} + \cdots\right]}_{\text{cont'd}}$$

$$+\underbrace{\sum_{i=1}^{n_h}\theta_i\sum_{r=1}^{P}\psi_i^{(r)}\sum_{q_1,\cdots,q_r=1}^{n_y}\prod_{s=1}^{r}\theta_{i,q_s+n_u+1}\left[h_0 + \sum_{k_1=0}^{\infty} h_1(k_1)\sum_{\ell=1}^{n} e^{j2\pi f_\ell(k-q_s-k_1)} + \cdots\right.}_{\text{pure output term}}$$

$$\underbrace{\left. \cdots + \sum_{k_1,\cdots k_n=0}^{\infty} h_n(k_1,\cdots,k_n)\prod_{i=1}^{n}\sum_{\ell=1}^{n} e^{j2\pi f_\ell(k-q_s-k_i)} + \cdots\right]}_{\text{cont'd}} \tag{30}$$

The n-th order GFRF can now be obtained by equating coefficients of $e^{j2\pi(f_1+\cdots+f_n)k}$ after collecting the relevant terms associated with $e^{j2\pi(f_1+\cdots+f_n)k}$ in equations (29) and (30).

The procedure described above outlines the basic steps involved in applying the harmonic expansion method to extract the required GFRF's from a given nonlinear system. To simplify the arithmetic operations, a so called *extraction operator*, $\varepsilon_n\{\cdot\}$, is defined to represent the above procedure [28]. It can be shown that $\varepsilon_n\{\cdot\}$ is a linear operator in the sense that results produced by the operation follow the superposition principle [19].

In the second stage of the harmonic expansion method, a transformation of equation (27) to the frequency domain is required. But since equation (27) consists of four categories of terms, these can easily be dealt with by applying $\varepsilon_n\{\cdot\}$ to each in turn. Furthermore, because of the linearity property of the extraction operator, only the product terms associated with the input and/or output variables (including the constant term) will be directly involved in the transformation. Thus, the amount of algebraic manipulation required can be greatly reduced. Details of this are given below. Finally, equation (27) can be transformed to a frequency domain representation by combining all the individual results.

Consider the transformation of each individual term in equation (27). For the constant term, the n-th order transformation can be obtained directly by applying the extraction operator to yield

$$\varepsilon_n\left\{\sum_{i=1}^{n_h}\theta_i\sigma(b_i)\right\} = \delta(f_1,\cdots,f_n)\sum_{i=1}^{n_h}\theta_i\sigma(b_i), \qquad n\geq 1 \tag{31}$$

where $\delta(f_1,\cdots,f_n)$ is defined by

$$\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\delta(f_1,\cdots,f_n)df_1\cdots df_n = \begin{cases}1 & \text{if } f_1=f_2\cdots=f_n=0\\ 0 & \text{otherwise}\end{cases} \tag{32}$$

Likewise, for the pure input term, the n-th order transformation gives

$$\varepsilon_n\left\{\prod_{r=1}^{r}u(k-p_r)\right\}=\begin{cases}e^{-j2\pi(f_1p_1+\cdots+f_np_n)} & \text{if } n=r,\ n\geq 1\\ 0 & \text{if } n\neq r\end{cases} \tag{33}$$

When the extraction operator is applied to the pure output term, the transformation $\varepsilon_n\left\{\prod_{s=1}^{r}y(k-q_s)\right\}$, however, needs to be computed in a recursive manner [8]. For notional convenience, define the auxiliary variable $H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)$ as the contribution to the n-th order GFRF that is generated by the r-th degree nonlinearity in the output, that is

$$H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)\triangleq\varepsilon_n\left\{\prod_{s=1}^{r}y(k-q_s)\right\},\qquad n\geq 0,\ r\geq 1 \tag{34}$$

Using the results from [28], $H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)$ can be written as

$$\begin{aligned}H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)&=H_0H_{n,r-1}(f_1,\cdots,f_n;q_1,\cdots,q_{r-1})\\&+\sum_{j=1}^{n}H_j(f_1,\cdots,f_j)H_{n-j,r-1}(f_{j+1},\cdots,f_n;q_1,\cdots,q_{r-1})e^{-j2\pi(f_1+\cdots+f_j)q_r},\qquad n\geq 0,\ r\geq 1\end{aligned} \tag{35}$$

Notice that $H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)$ in equation (35) has been recursively defined and can be further expanded to yield

$$\begin{aligned}H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)&=H_0^{r-1}H_n(f_1,\cdots,f_n)\sum_{s=1}^{r}e^{-j2\pi(f_1+\cdots+f_n)q_s}\\&+\sum_{s=1}^{r-1}H_0^{s-1}\sum_{j=1}^{n-1}H_j(f_1,\cdots,f_j)H_{n-j,r-s}(f_{j+1},\cdots,f_n;q_1,\cdots,q_{r-s})e^{-j2\pi(f_1+\cdots+f_j)q_{r-s+1}},\quad n\geq 0,\ r\geq 1\end{aligned} \tag{36}$$

after $(r-1)$ self-iterations. The recursive relationship given in equation (36) finishes at $r=1$ with

$$H_{n,1}(f_1,\cdots,f_n;q_1)=H_n(f_1,\cdots,f_n)e^{-j2\pi(f_1+\cdots+f_n)q_1},\qquad n\geq 1 \tag{37}$$

and $H_{0,r}$ $(r\geq 1)$ is defined as

$$H_{0,r}\triangleq\varepsilon_0\left\{\prod_{s=1}^{r}y(k-q_s)\right\}=H_0^r,\qquad r\geq 1 \tag{38}$$

where H_0 can be physically interpreted as the steady state response of the network. The value of H_0 can be determined by solving the input-output relationship of the network in equation (27) in the limiting case when $k\to\infty$ and $u(k)\equiv 0$ such that

$$H_0=\sum_{i=1}^{n_h}\theta_i\sigma(b_i)+\sum_{i=1}^{n_h}\theta_i\sum_{r=1}^{P}\psi_i^{(r)}\sum_{q_1,\cdots,q_r=1}^{n_y}\prod_{s=1}^{r}\theta_{i,q_s+n_u}H_0^r \tag{39}$$

Equation (39) is a nonlinear algebraic equation which can be solved numerically using some of the well known numerical techniques such as the Laguerre method. In general, this will give a number of real and complex solutions. For the sake of physical interpretability, all complex roots will be

discarded. Unfortunately there is no known systematic mechanism to determine which of the remaining solutions to use for a particular model with given specifications. Therefore, all the real solutions must be treated as candidates which are considered in turn in the computations of the GFRF's. The results of the computations must then be assessed on an individual basis by using known *a priori* knowledge of the underlying system.

By using the alternative interpretation expressed in equation (34), the definition of $H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)$ infers

$$H_{n,0}=\varepsilon_n\left\{\sum_{i=1}^{n_h}\theta_i\sigma(b_i)\right\}=\begin{cases}\displaystyle\sum_{i=1}^{n_h}\theta_i\sigma(b_i) & \text{if } n=0\\ \delta(f_1,\cdots,f_n)\displaystyle\sum_{i=1}^{n_h}\theta_i\sigma(b_i) & \text{if } n\geq 1\end{cases} \tag{40}$$

Using the product identity derived from the basic properties of $\varepsilon_n\{\cdot\}$ given in [28], the cross product contribution $\varepsilon_n\left\{\prod_{v=1}^{r-l}u(k-p_v)\prod_{s=1}^{l}y(k-q_s)\right\}$ can be factorised into products of pure input and pure output term contributions in the following form

$$\varepsilon_n\left\{\prod_{v=1}^{r-l}u(k-p_v)\prod_{s=1}^{l}y(k-q_s)\right\}\equiv\varepsilon_{r-l}\left\{\prod_{v=1}^{r-l}u(k-p_v)\right\}\varepsilon_{n-r+l}\left\{\prod_{s=1}^{l}y(k-q_s)\right\},\qquad n\geq 1 \tag{41}$$

From equations (33) and (34), the cross product term contribution can therefore be written as

$$\varepsilon_n\left\{\prod_{v=1}^{r-l}u(k-p_v)\prod_{s=1}^{l}y(k-q_s)\right\}$$
$$=e^{-j2\pi(f_1p_1+\cdots+f_{r-l}p_{r-l})}H_{n-r+l,l}(f_{r-l+1},\cdots,f_n;q_1,\cdots,q_l),\qquad n\geq 1 \tag{42}$$

By combining the results of the transformation for each of the constituent terms in equation (27), the n-th order GFRF can be obtained by substituting equations (33), (34) and (42) into equation (27) to yield

$$H_n(f_1,\cdots,f_n)=\underbrace{\delta(f_1,\cdots,f_n)\sum_{i=1}^{n_h}\theta_i\sigma(b_i)}_{\text{constant term contribution}}+\underbrace{\sum_{i=1}^{n_h}\theta_i\psi_i^{(n)}\sum_{p_1,\cdots,p_r=0}^{n_u}\prod_{v=1}^{n}\theta_{i,p_v+1}e^{-j2\pi(f_1p_1+\cdots+f_np_n)}}_{\text{pure input term contribution}}$$
$$+\underbrace{\sum_{i=1}^{n_h}\theta_i\sum_{r=1}^{P}\psi_i^{(r)}\sum_{q_1,\cdots,q_r=1}^{n_y}\prod_{s=1}^{r}\theta_{i,q_s+n_u}H_{n,r}(f_1,\cdots,f_n;q_1,\cdots,q_r)}_{\text{pure output term contribution}}$$

$$\underbrace{+\sum_{i=1}^{n_h}\theta_i\sum_{r=2}^{P}\psi_i^{(r)}\sum_{l=1}^{r-1}\binom{r}{l}\left[\sum_{p_1,\cdots,p_{r-l}=1}^{n_u}\prod_{v=1}^{r-l}\theta_{i,p_v}\right]e^{-j2\pi(f_1p_1+\cdots+f_{r-l}p_{r-l})}}_{\text{cross product term contribution}}$$

$$\underbrace{\times\sum_{q_1,\cdots,q_l=1}^{n_y}\prod_{s=1}^{l}\theta_{i,q_s+n_u}H_{n-r+l,l}(f_{r-l+1},\cdots,f_n;q_1,\cdots,q_l)}_{\text{(cont'd)}},\qquad n\geq 1 \tag{43}$$

It is interesting to note that when equation (27) is transformed into the GFRF's, following equation (20), only the pure n-th degree polynomial input terms contribute to the n-th order GFRF. Therefore, the choice of the maximum degree, P, in the polynomial in equation (13) has a direct impact on the richness of the higher order frequency response functions of the underlying network. In general, the larger the value of P, the longer the (truncated) Taylor series expansion of the activation function is and thus the computed GFRF's will be more accurate. Unfortunately, the determination of the coefficients in the Taylor expansion is computationally intensive for large P because of the computation of the required higher order derivatives. Nevertheless, experience suggests that satisfactory results for low order (up to third) GFRF's can generally be obtained with a typical choice of P in the range from 6 to 10 which is computationally manageable on a standard workstation. An account of the effects of the choice of the largest polynomial degree on the computation of the Volterra kernels has been discussed in [27].

Finally, combining equation (43) and equation (36) and rearranging terms gives

$$\begin{aligned}H_n(f_1,\cdots,f_n)=\Bigg\{&\delta(f_1,\cdots,f_n)\sum_{i=1}^{n_h}\theta_i\sigma(b_i)+\sum_{i=1}^{n_h}\theta_i\psi_i^{(n)}\sum_{p_1,\cdots,p_r=1}^{n_u}\prod_{v=1}^{n}\theta_{i,p_v}e^{-j2\pi(f_1p_1+\cdots+f_np_n)}\\
&+\sum_{i=1}^{n_h}\theta_i\sum_{r=2}^{P}\psi_i^{(r)}\sum_{q_1,\cdots,q_r=1}^{n_y}\prod_{s=1}^{r}\theta_{i,q_s+n_u}\Bigg[\sum_{s=1}^{r-1}H_0^{s-1}\sum_{j=1}^{n-1}H_j(f_1,\cdots,f_j)\\
&\times\sum_{j=1}^{n-1}H_j(f_1,\cdots,f_j)H_{n-j,r-s}(f_{j+1},\cdots,f_n;q_1,\cdots,q_{r-s})e^{-j2\pi(f_1+\cdots+f_j)q_{r-s+1}}\Bigg]\\
&+\sum_{i=1}^{n_h}\theta_i\sum_{r=2}^{P}\psi_i^{(r)}\sum_{l=1}^{r-1}\binom{r}{l}\Bigg[\sum_{p_1,\cdots,p_{r-l}=1}^{n_u}\prod_{v=1}^{r-l}\theta_{i,p_v}e^{-j2\pi(f_1p_{r-l}+\cdots+f_{r-l}p_{r-l})}\Bigg]\\
&\times\Bigg[\sum_{q_1,\cdots,q_l=1}^{n_y}\prod_{s=1}^{l}\theta_{i,q_s+n_u}H_{n-r+l,l}(f_{r-l+1},\cdots,f_n;q_1,\cdots,q_l)\Bigg]\Bigg\}\\
&\times\left\{1-\sum_{i=1}^{n_h}\theta_i\sum_{r=1}^{P}\psi_i^{(r)}H_0^{r-1}\sum_{q_1,\cdots,q_r=1}^{n_y}\prod_{s=1}^{r}\theta_{i,q_s+n_u}\sum_{s=1}^{r}e^{-j2\pi(f_1+\cdots+f_n)q_s}\right\}^{-1},\qquad n\geq 1\end{aligned} \tag{44}$$

The analytical expression in equation (44) is order-recursive. The recursive equation (44) can be taken, together with equation (36), as the algorithm for computing the n-th order GFRF of a neural network model.

In practice, the computation of $H_n(f_1,\cdots,f_n)$ is implemented by encoding the recursive algorithm expressed in equations (36) and (44) on a digital computer with the initial conditions defined above so that the n-th order GFRF for an ANN can be evaluated automatically. Because of the order-recursive nature of the algorithm, the computations of the higher order GFRF's are always based on the results obtained for the lower order functions. For the vast majority of applications computation of just the first two or three GFRF's is sufficient to characterise the main frequency domain features although the higher order GFRF's do provide additional information in some cases. To minimise the computational burden, Wray and Green [26] suggested that finite polynomial output functions can be used as alternative nodal functions to the conventional sigmoid or hyperbolic tangent activation functions considered above. But this can degrade the approximation capabilities of the multilayered perceptron network and therefore errors can easily be introduced.

5. Numerical Examples

Nonlinear system identification is one of the application areas of ANN. The greatest advantage of using ANN's lies in the fact that they provide a standard architectural framework which can be used as a universal approximator to provide an approximation to the underlying system dynamics. However, the advantage of such flexibility is often offset by the loss of structural information. As a result, over-fitting is not uncommon. In addition, it is commonly believed that the training of MLP's is highly problem dependent. In the absence of detailed *a priori* knowledge about the underlying system specifications, optimal training can be difficult to achieve without an extensive trail-and-error study to select the appropriate network configuration and algorithmic settings such as the learning rate and the momentum constant. Furthermore, the multi-modal nature of the (non-convex) cost function means that local minima are a common problem when the widely adopted back propagation algorithm is employed. These are some of the subtle issues which have been documented in numerous studies [15][14].

In the following subsections, two examples will be given to illustrate the computation of the GFRF's for ANN's and to demonstrate how the network configuration and training influence these results. In all the simulated systems, 10,000 system input-output data samples were

generated. In each case, an MLP with hyperbolic tangent activation functions was then trained extensively using the conventional generalised delta rule for 100 passes. Therefore, a total of one million ($=10,000\times100$) system input-output data were actually scanned to allow sufficient training. Whenever appropriate, the learning rate and momentum constant were manually fined tuned in order to obtain the best possible training results. Before a training session was started, all the network weights were randomly initialised to some small arbitrary values in the range $[-1/2, 1/2]$ to avoid numerical degeneracy.

Once a training session was completed, the first 12, 8 and 6 terms in the Taylor series expansion of the activation function were used, respectively, in the computations of the first, second and third order GFRF's of the trained network. During the computation of the GFRF's, all the possible (real valued) steady state responses were considered and only the one with sensible supporting arguments was adopted. Obviously while this can easily be determined in simulated systems it may present practical difficulties in selecting the proper solution when real data sets are encountered. Further investigation on this issue is necessary to enhance the usefulness of the algorithm. To assess the quality of the trained network, the mean squared errors were also computed (in decibels) using the 10000 output data in the 100-th training pass thus ensuring all the transitory effects were completely ignored.

5.1. Example 1: Feedforward Network for Linear System Modelling

In the first example, the effects of overfitting an MLP model was investigated by considering the identification of a simple linear system which is defined by the first order ARX process described as

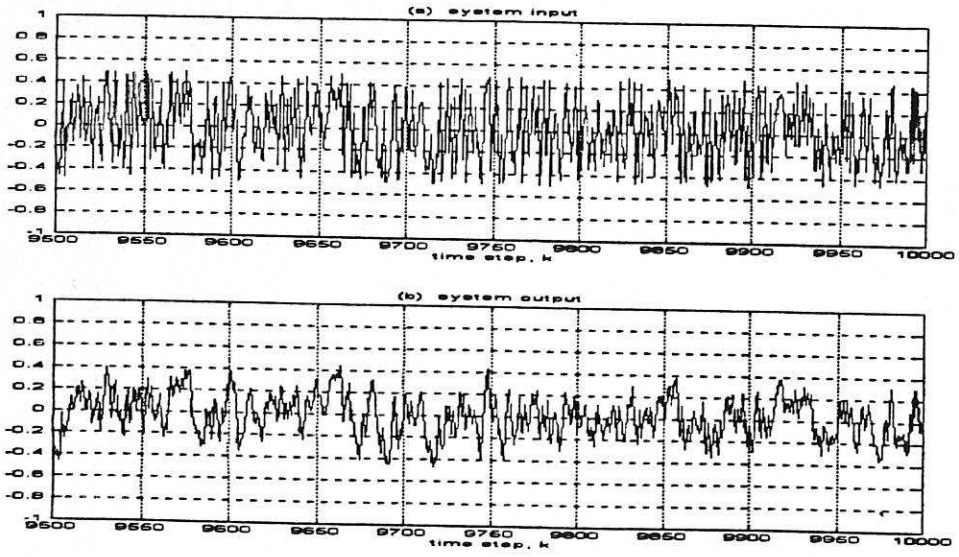
$$y(k)=0.5u(k-1)+0.6y(k-1) \tag{45}$$

where $\{u(k)\}$ is a uniformly distributed random sequence in $[-1/2, 1/2]$. This example was specifically chosen to demonstrate how easy it is to overfit a neural network.

Assume that the physical structure of the underlying system is not known in advance. This would be the situation that exists for most real data sets. An 18-hidden node, three layered feedforward network with an input vector $\mathbf{x}(k)=[u(k-1), u(k-2), \cdots, u(k-10)]^T$ was blindly used to represent this system. The network was trained using the generalised delta rule [24][21] with the learning rate chosen to be 0.05 and a momentum constant of 0.2. A mean squared error of

-32.44 dB was achieved in the final pass of the training process. A set of traces from the training process are given in Figure 2 and these show that the network accurately predicts the system output but there is no indication that the network is grossly overfitted. A comparison of the true transfer function of the linear system in equation (45) and the first order GFRF of the trained network is shown in Figure 3 which shows that these are almost coincident. The trained network clearly approximates the underlying linear system within the entire frequency range.

During the experiment, the network was manually optimised by varying the network topology in terms of the number of hidden nodes and the largest input lag used in the input node assignments in order to obtain the best possible GFRF's which matched those of the true linear system. Similarly the learning rate and momentum constant were fine tuned during the training process. It should however be noticed that in this particular example the linear system has been deliberately modelled by an overfitted model structure. This results in a model mismatch in the sense that the recursive linear system has been represented by a feedforward nonlinear model. Thus, it can be expected that despite the attempt to optimise the system representation, the resulting model exhibits non-vanishing higher order nonlinear dynamics which is reflected in the second and third order GFRF's.



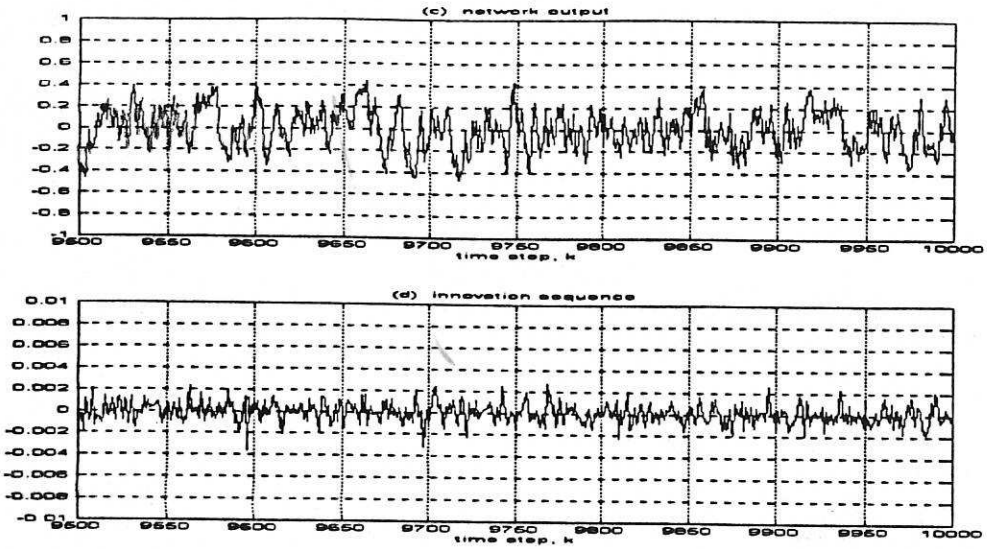


Figure 2 Traces of the training process for example 1; (a) system input, (b) system output, (c) network one step ahead predicted output and (d) innovation sequence of the network.

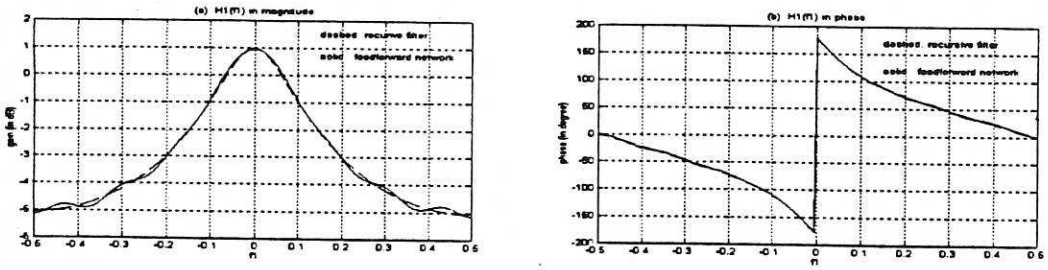


Figure 3 Transfer function of the linear recursive filter (solid line) versus the first order GFRF of the trained network model (dashed line) used for example 1; (a) gain and (b) phase.

The second and third order GFRF's of the trained network are shown in Figures 4 and 5, respectively. Because the real system equation (45) is linear both these should ideally be zero. The estimates reveal that both the second and third order GFRF's exhibit non-zero responses but these are small in absolute gain levels. This suggests that the model in question exhibits some degree of nonlinearity which of course cannot possibly be generated by any kind of linear system. In other words, this is an indication that the underlying (linear) system defined in equation (45) has been overfitted.

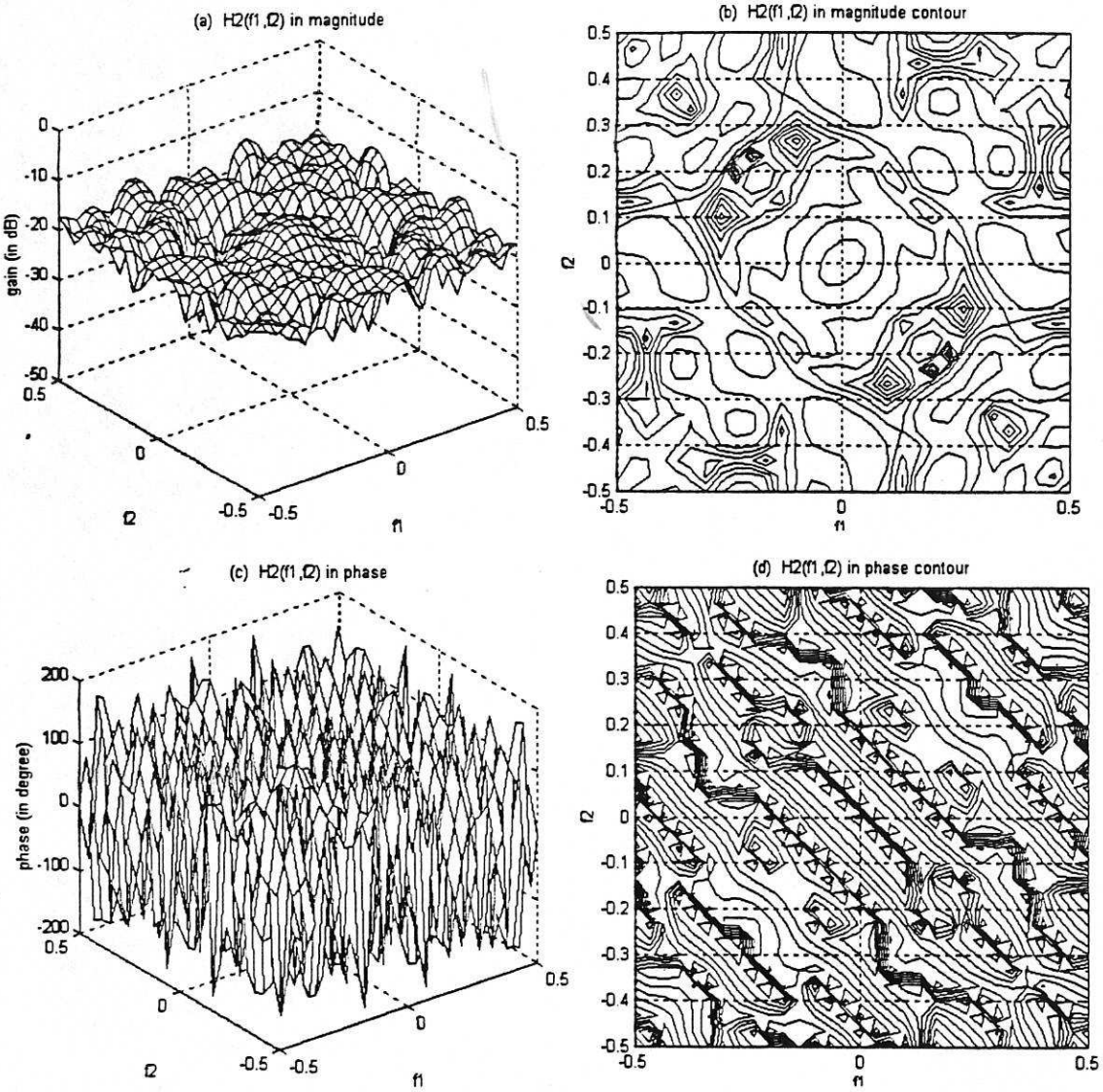


Figure 4 Second order GFRF of the trained network model for example 1; (a) gain in dB, (b) gain contour, (c) phase in degrees and (d) phase contour.

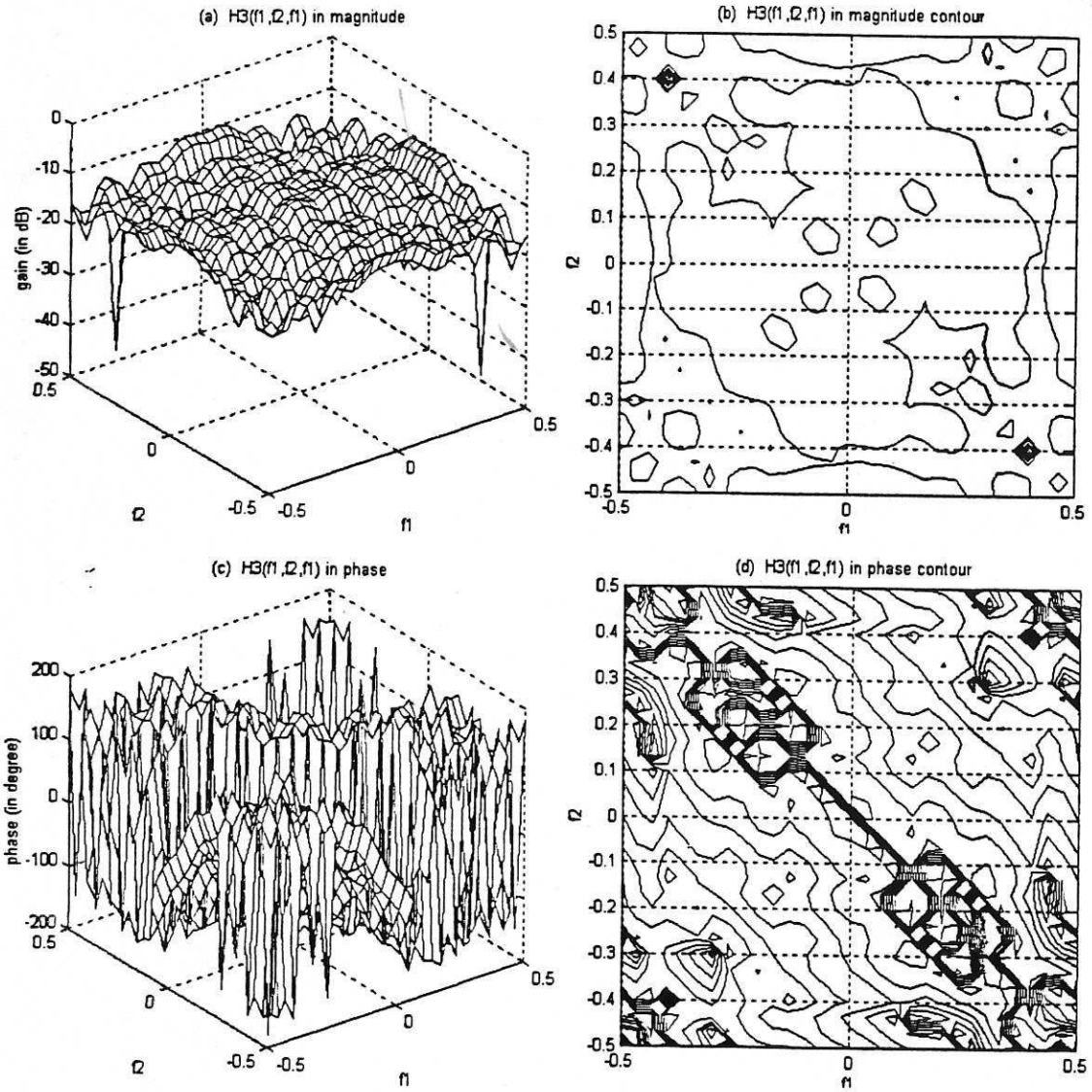


Figure 5 Third order GFRF with $f_3 = f_1$ of the trained network model for example 1; (a) gain in dB, (b) gain contour, (c) phase in degrees and (d) phase contour.

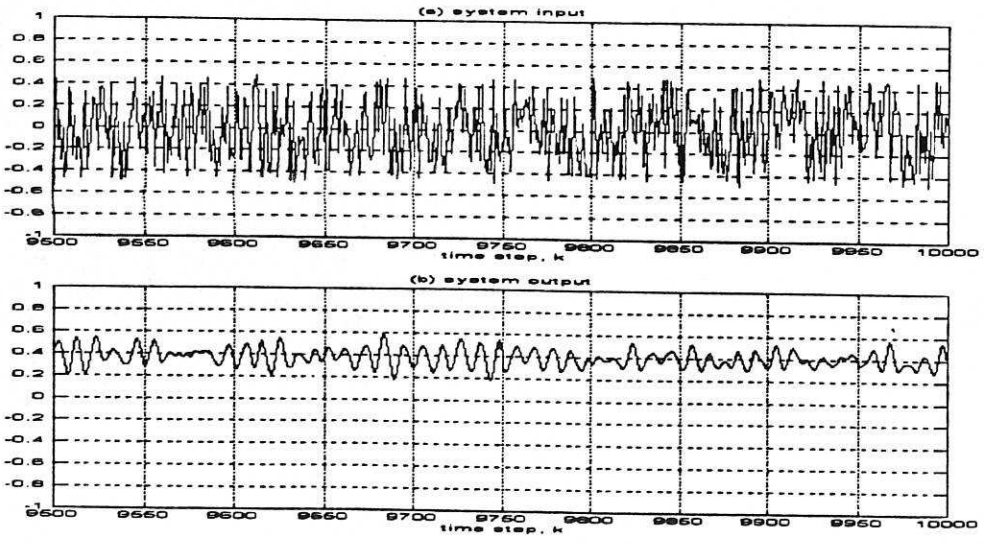
5.2. Example 2: A Recurrent Network for Polynomial NARX System Modelling

As a second example consider a nonlinear system defined by the nonlinear difference equation

$$y(k) = 0.13597 + 0.06149u(k-1) + 1.6021y(k-1) - 0.94726y(k-2) - 0.013829y^2(k-1) - 0.0025225y^3(k-1) \quad (46)$$

where $\{u(k)\}$ is a uniformly distributed random sequence in $[-1/2, 1/2]$. A three layered recurrent network with an input vector correctly assigned as $\mathbf{x}(k)=[u(k-1), y(k-1), y(k-2)]^T$ was used to represent this system. This system has previously been studied in the identification of nonlinear circuits and the first and second order GFRF's can be derived analytically [8].

In this experiment, the influence of the network model structure was investigated by comparing the dynamical responses of two networks both in the time and the frequency domains. This example was designed to show the sensitivity of the trained network GFRF's, which characterise the dynamical properties of the network to small changes in the network structure. The experiment was carried out initially for an 11-hidden node MLP and then for a 12-hidden node MLP while keeping the rest of the experimental conditions unchanged. In both cases, a generalised delta rule with a learning rate of 0.05 and momentum constant of 0.3 were used after an extensive trial-and-error search to optimise these parameters. Figure 6 depicts a set of traces of the training results for both the 11 and 12-hidden node MLP's. The former achieved a mean squared error of -68.08 dB whereas the later achieved a remarkably close -68.31 dB. The one step ahead predictions, Figure 6(c) and Figure 6(d), of these networks and the associated innovation sequences as shown in Figures 6(e) and 6(f) are virtually identical and both provide excellent predictions of the system output. Based on these traces, it would be quite impossible to differentiate the relative merits of these trained networks.



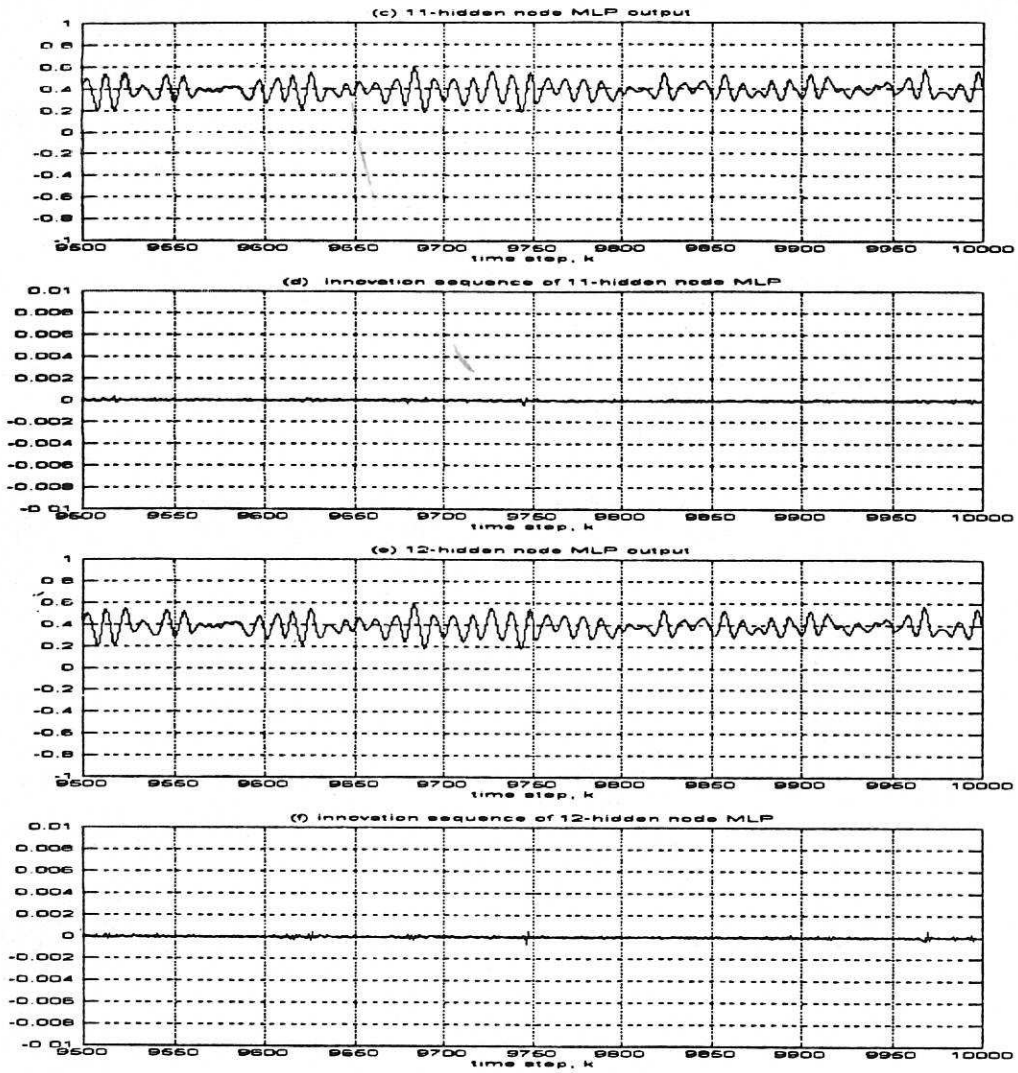


Figure 6 Traces of the training process for example 2; (a) system input, (b) system output, (c) network one step ahead predicted output of the 11-hidden node MLP, (d) innovation sequence of the 11-hidden node MLP, (e) network output of the 12-hidden node MLP and (f) innovation sequence of the 12-hidden node MLP.

The true GFRF's associated with the model of equation (46) are shown in Figures 7(a), 7(b) and Figures 8(a) to 8(d) [8].

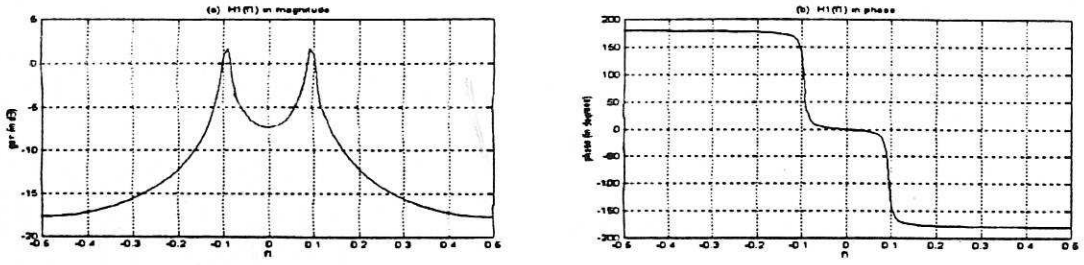


Figure 7 First order GFRF of the system described by equation (46) for example 2; (a) gain in dB and (b) phase in degrees.

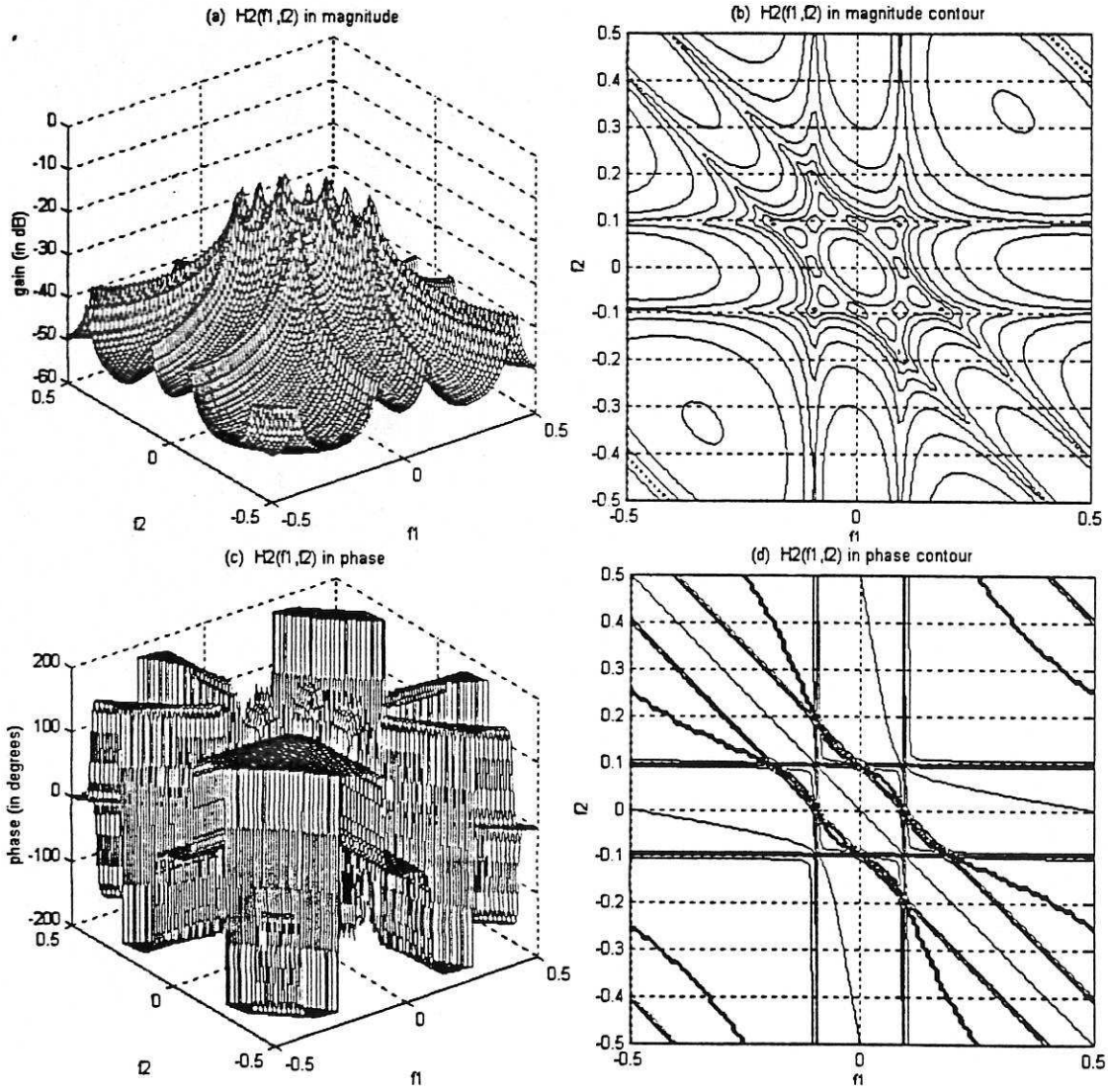


Figure 8 Second order GFRF of the system described by equation (46) for example 2; (a) gain in dB, (b) gain contour, (c) phase in degrees and (d) phase contour.

When the trained networks are mapped into the frequency domain using the results derived above there are considerable differences between all orders of GFRF's. The 12 hidden node MLP comes closest to reproducing the correct frequency response effects. But notice how the resonant frequencies in the $H_1(f_1)$ plots of the two networks Figures 9(a), 9(b) and Figures 12(a), 12(b) move despite the tiny difference in the mean squared errors of -68.08 dB compared to -68.31 dB. A comparison of $H_2(f_1, f_2)$'s (Figures 10(a) to 10(d) and Figures 13(a) to 13(d)) and $H_3(f_1, f_2, f_3)$'s (Figures 11(a) to 11(d) and Figures 14(a) to 14(d)) of the two networks also show a large apparent sensitivity of these functions to the slight changes between the networks. This is despite the fact that the performance in the time domain of both the 11 and 12 node networks are virtually identical.

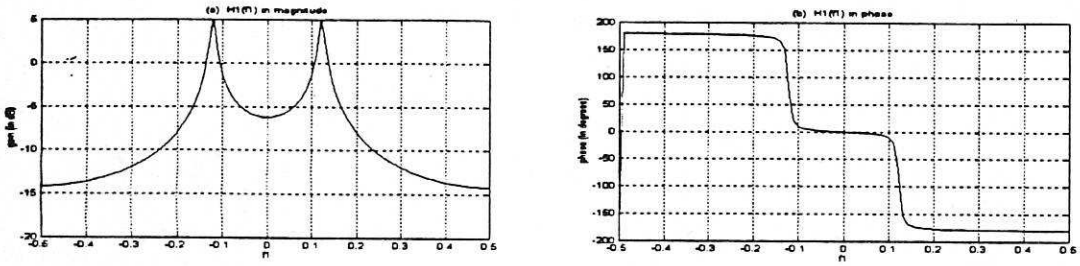


Figure 9 First order GFRF of the 11-hidden node MLP network model for example 2; (a) gain in dB and (b) phase in degrees.

We have carefully rechecked these results to confirm that they are correct. Other simulations provided results which were also sensitive to minor changes in the network training. The approximation in equation (13) will be an influence on the estimates but our results suggest that the effects are probably caused by overparameterisation. That is the networks can provide good time domain predictions but they have not captured the underlying dynamics of the system as defined by the GFRF's.

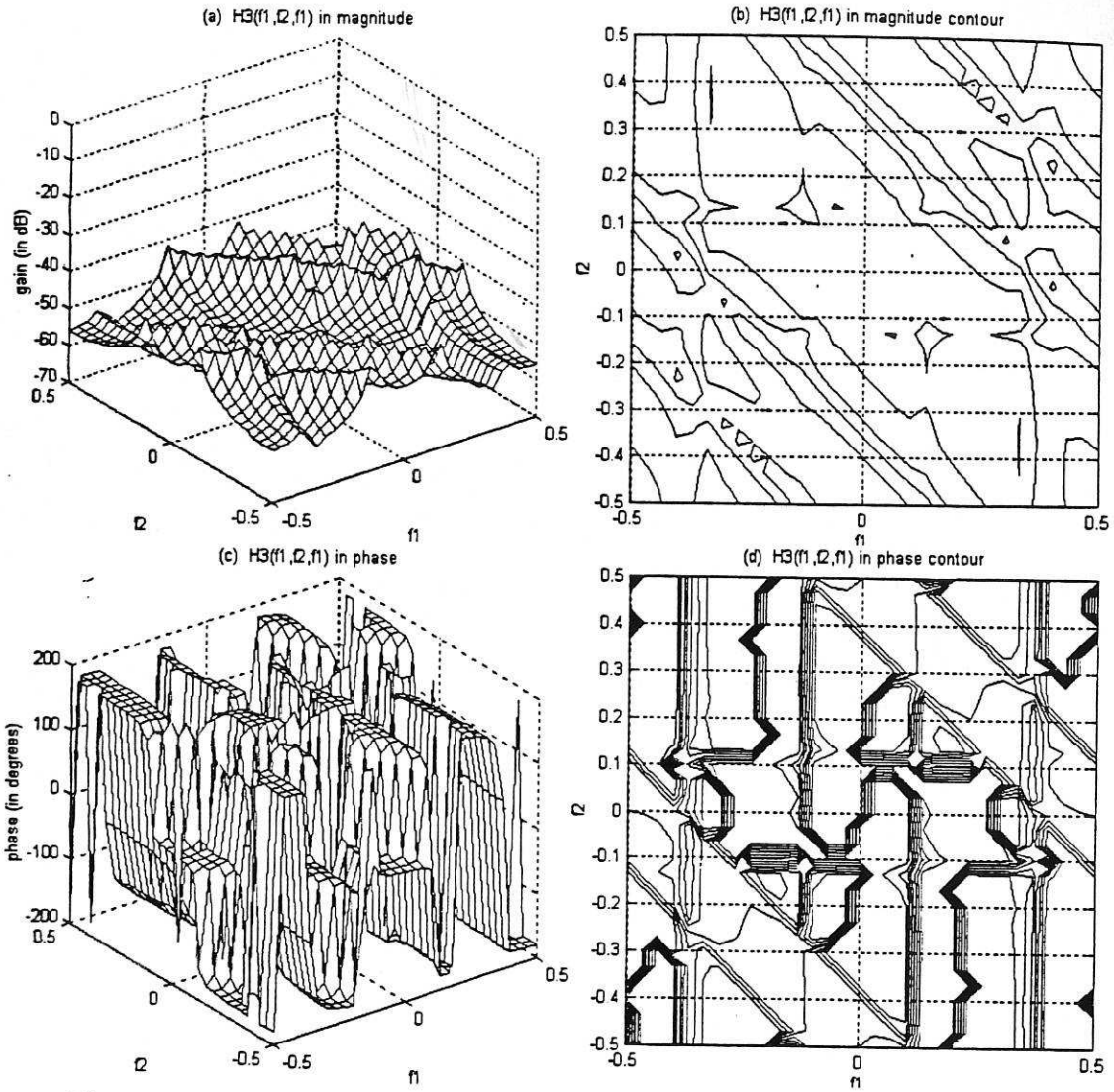


Figure 11 Third order GFRF (with $f_3 = f_1$) of the 11-hidden node network model for example 2; (a) gain in dB, (b) gain contour, (c) phase in degrees and (d) phase contour.

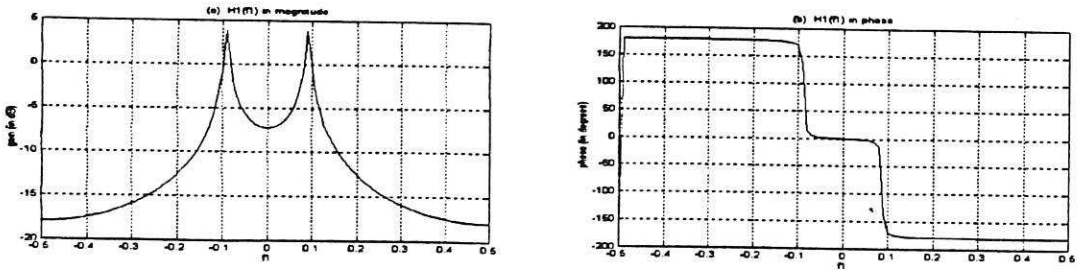


Figure 12 First order GFRF of the 12-hidden node MLP network model for example 2; (a) gain in dB and (b) phase in degrees.

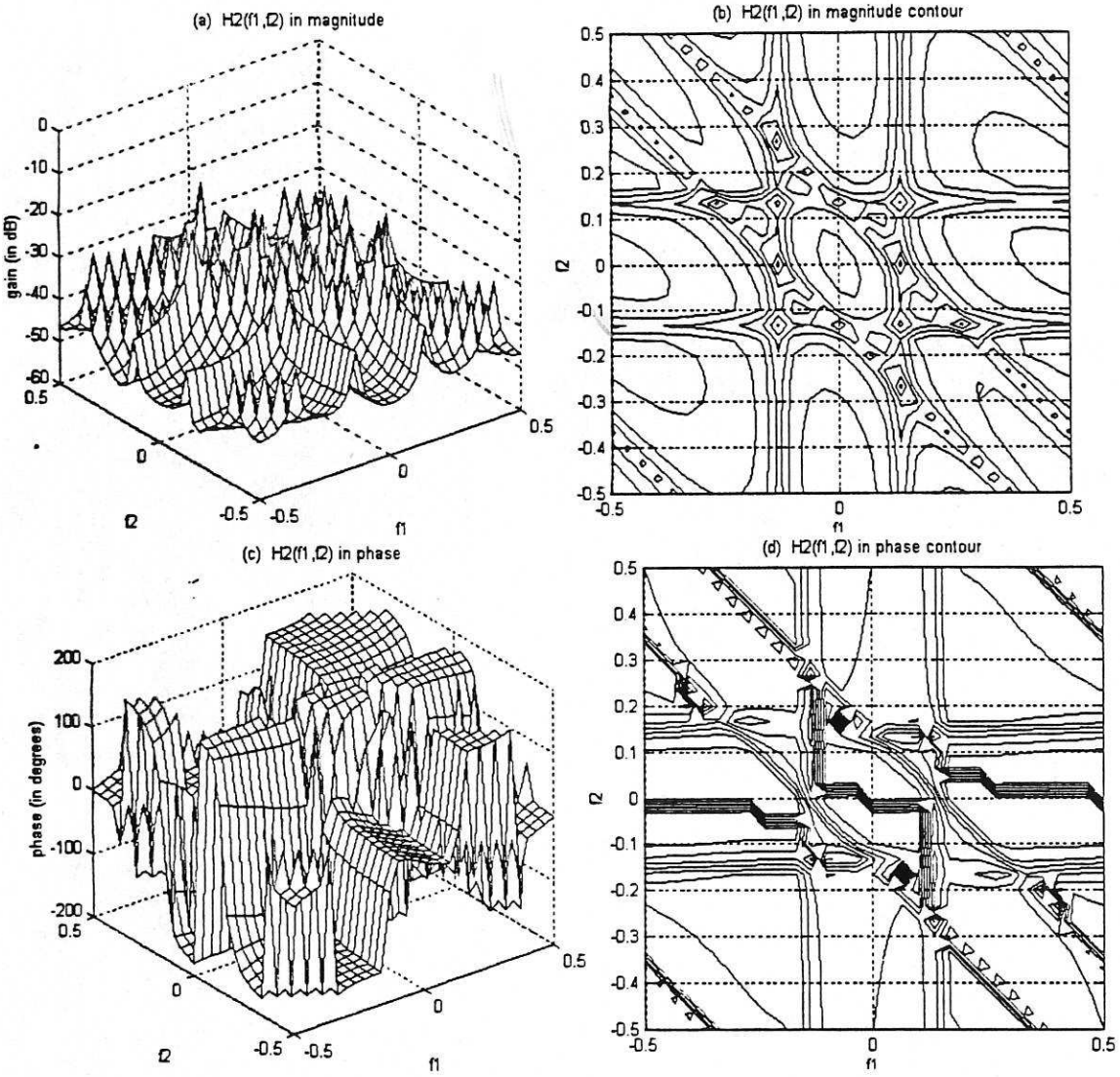


Figure 13 Second order GFRF of the 12-hidden node MLP network model for example 2; (a) gain in dB, (b) gain contour, (c) phase in degrees and (d) phase contour.

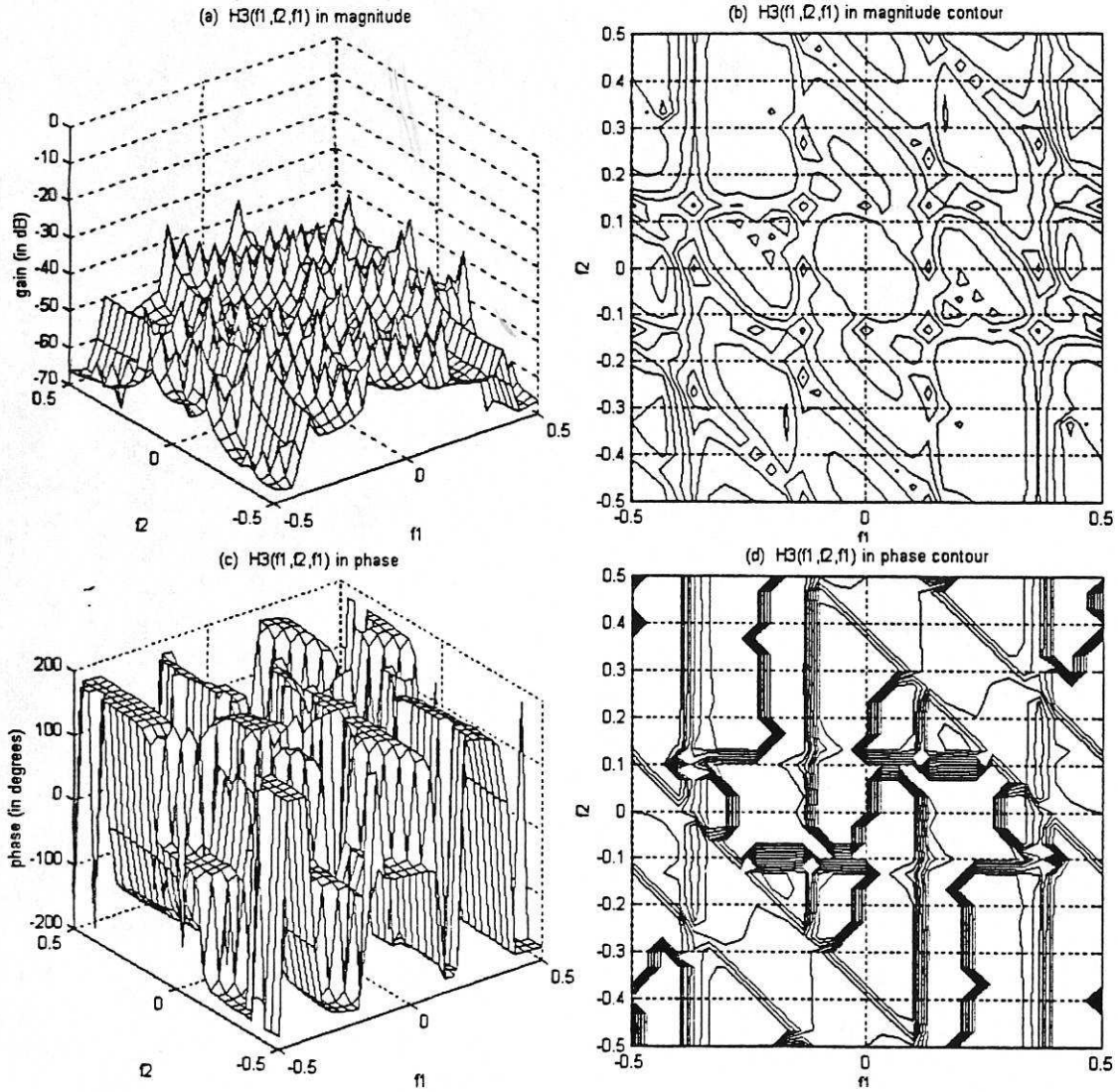


Figure 14 Third order GFRF (with $f_3 = f_1$) of the 12-hidden node network model for example 2; (a) gain in dB, (b) gain contour, (c) phase in degrees and (d) phase contour.

6. Conclusions

A new algorithm has been derived which maps feedforward and recurrent networks into the generalised frequency response functions (GFRF's). This allows the trained time domain neural network model to be transformed into the frequency domain and provides insight into the internal dynamic characteristics of the network.

However, the simulation results suggest that the form of GFRF's obtained are sensitive to both the user selectable training parameters and the network topology. Even slight changes in the mean squared errors, for example, -68.08 dB compared to -68.31 dB can provide important changes in the GFRF's. This suggests that selecting networks based on the mean squared error performance will be very difficult if the objective is to construct a model which characterises the dynamics of the underlying system rather than to provide good time domain predictions. Initial investigations suggest that these effects may be caused by overparameterisation but further studies are underway to establish if the results reported here are representative and what causes them.

Acknowledgements

SAB gratefully acknowledges that part of this work was supported by EPSRC.

References

[1] Bedrosian, E. and Rice, S. O. (1971). The output properties of Volterra systems (nonlinear system with memory) driven by harmonic and Gaussian inputs. *Proceedings of the IEEE*, **59**, 1688-1707.

[2] Billings, S. A. and Chen, S. (1989). Extended model set, global data and threshold model identification of severely non-linear systems. *International Journal of Control*, **50**(5), 1897-1923.

[3] Billings, S. A., Chen, S. and Korenberg, M. J. (1989). Identification of MIMO nonlinear systems using a forward-regression orthogonal estimator. *International Journal of Control*, **49**(6), 2157-2189.

[4] Billings, S. A. and Tsang, K. M. (1989). Spectral analysis for nonlinear systems, Part 1: parametric nonlinear spectral analysis. *Journal of Mechanical Systems and Signal Processing*, **3**(4), 319-339.

[5] Billings, S. A., Tsang, K. M. and Tomlinson, G. R. (1990). Spectral analysis for nonlinear systems, part 3 - case study examples. *Journal of Mechanical Systems and Signal Processing*, **4**(1), 3-21.

[6] Billings, S. A., Jamaluddin, H. B. and Chen, S. (1992). Properties of neural networks with applications to modelling nonlinear dynamical systems. *International Journal of Control*, **55**(1), 193-224.

[7] Billings, S. A. and Fung, C. F. (1995). Recurrent radial basis function networks for adaptive noise cancellation. *Neural Networks*, **8**(2), 273-290.

[8] Billings, S. A. and Zhang, H. (1995). Computation of nonlinear transfer functions when constant terms are present. *Mechanical Systems and Signal Processing*, **9**(5), 537-553.

[9] Boyd, S., Chua, L. O. and Desoer, C. A. (1984). Analytical foundations of Volterra series. *IMA Journal of Mathematical Control and Information*, **1**, 243-282.

[10] Brillinger, D. R. (1970). The identification of polynomial systems by means of higher order spectra. *Journal of Sound and Vibration*, **12**, 301-431.

[11] Chen, S., Billings, S. A. and Grant, P. M. (1990). Nonlinear system identification using neural networks, *International Journal of Control*, **51**(6), 1191-1212.

[12] Chua, L. O. and Ng, C. Y. (1979). Frequency domain analysis of nonlinear systems: general theory. *IEE Journal of Electronic Circuits and Systems*, **3**(4), 165-185.

[13] Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**, 303-314.

[14] Haykin, S. (1994). *Neural Networks: A comprehensive foundation*. New York: Macmillan College Publishing Company.

[15] Hertz, J., Krogh, A. and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley Publishing Company.

[16] Hunt, K. J. and Sbarbaro, D. (1991). Neural networks for nonlinear internal model control. *IEE Proceedings, Part D*, **138**, 431-438.

[17] Leontaritis, J. J. and Billings, S. A. (1985). Input-output parametric models for nonlinear systems, part 1: deterministic nonlinear systems. *International Journal of Control*, **41**(2), 303-328.

[18] Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, **1**(1), 4-27.

[19] Peyton Jones, J. C. and Billings, S. A. (1989). Recursive algorithm for computing the frequency response of a class of nonlinear difference equation models. *International Journal of Control*, **50**(5), 1925-1940.

[20] Peyton Jones, J. C. and Billings, S. A. (1993). Mean levels in nonlinear analysis and identification. *International Journal of Control*, **58**(5), 1033-1052.

[21] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533-536.

[22] Volterra, V. (1959). *Theory of Functionals and Integral and Integro-Differential Equations*. New York: Dover.

[23] Waibel, A. Hanazawa, T, Hinton, G. Shikano, K. and Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **37**, 328-339.

[24] Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in The Behavioural Sciences*. Ph.D. Dissertation, Harvard University.

[25] Wiener, N. (1958). *Nonlinear Problems in Random Theory*. New York: Technology Press.

[26] Wray, J. and Green, G. (1991). Analysis of networks that have learnt control problems. In *Proceedings of the IEEE Control 91* (pp. 261-265). Herriot Watt, UK.

[27] Wray, J. and Green, G. G. R. (1994). Calculation of the Volterra kernels of nonlinear dynamic systems using an artificial neural network. *Biological Cybernetics*, **71**, 187-195.

[28] Zhang, H., Billings, S. A. and Zhu, Q. M. (1995). Frequency response functions for nonlinear rational models. *International Journal of Control*, **61**(5), 1073-1097.